Microsoft

Microsoft Windows 2000 Server

Operating System

Developing Hardware Snapshot Providers for Whistler

White Paper. Updated for Whistler Server Beta 3

Abstract

Whistler adds support for Snapshots to the Microsoft® Windows® XP operating system. This White Paper describes the design issues involved for hardware vendors who wish to add a Snapshot Provider into this infrastructure.

Note that there are substantial changes from the provider white paper documentation provided at Whistler Beta 1.

EXHIBIT

tabbles

A

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

0999

INTRODUCTION

Definition Of Terms

Auto-Release	A snapshot that is deleted immediately after the Requestor exit or detach. Local (non-SAN) backup software snapshots are auto-release.
Copy-on-write	A snapshot created by saving only the differences to the original volume. See snapshot data.
Deleted Snapshot	Deleted snapshots are not accessible at all and must not be made accessible at any time in the future. Deleted snapshots may still occupy resources (for example space in a software storage File.)
Exposed Snapshot	Accessible in the Windows name space via drive letter, mount point, and/or network share.
GUID	Many of the ID numbers used in VSS, (and in many of Microsoft's technologies) are GUIDs – Globally Unique Identifiers. The CoCreateGuid() function creates a GUID, a globally unique 128-bit integer. Use the CoCreateGuid() function whenever an absolutely unique persistent identifier in a distributed environment is needed. To a very high degree of certainty, this function returns a unique value – no other invocation, on the same or any other system (networked or not) should ever return the same value. GUIDs should **not** be re-used. They are intended to be unique identifiers.
Hardware Snapshot Provider	Hardware providers consist of a user-mode component that works in conjunction with a hardware storage adapter or external RAID storage subsystem. I/Os are intercepted by the hardware (not software) which instantiates a snapshot LUN.
Hidden Snapshot	Not accessible in the Windows name space. Accessed only via //?/GLOBALROOT or otherwise masked.
Legacy Application	An application that does not include a writer and does not observe the snapshot synchronization protocol.
Managed Snapshot	A snapshot that can be queried via VSS. In particular, the original volume, snapshot set and snapshot set

	membership can be determined.
Multi-layer Snapshot	More than one snapshot exists for a specific original volume.
Original Volume	The volume from which a snapshot is derived.
Persistent Snapshot	Existing beyond a clean shutdown or hard crash of the operating system.
Plex	A snapshot initially created by mirroring. This special case of snapshot data that represents a snapshot without the need for the original volume data.
Requestor	A process that requests that one or more snapshot sets be taken of one or more original volumes. A backup application is a good example.
Snapshot	A read-only point-in-time replica of original volume state. Each snapshot is keyed by a persistent GUID.
Snapshot Data	A set of deltas and configuration information, which when applied to the original volume, comprise a snapshot.
Snapshot Provider	A software snapshot provider or a hardware snapshot provider. Providers own the snapshot data and instantiate the snapshot.
Snapshot Set	A set of snapshots with a common SnapShotSet identifier. Snapshot Sets are a collection of snapshots that were created at the same time. This is keyed by a persistent GUID.
Snapshot Storage File(s)	Resources consumed by a software copy-on-write software provider
Snapshot Storage Volume	A volume that holds snapshot storage files for a copy-on-write software provider
Software Snapshot Provider	Software providers are implemented as a user-mode component plus a kernel-mode driver. I/Os are intercepted by the driver that instantiates snapshot volumes. The driver may be a 'storage filter driver' or a component of a volume manager.
Transporting	Moving a managed persistent snapshot between hosts; this usually occurs on a SAN.

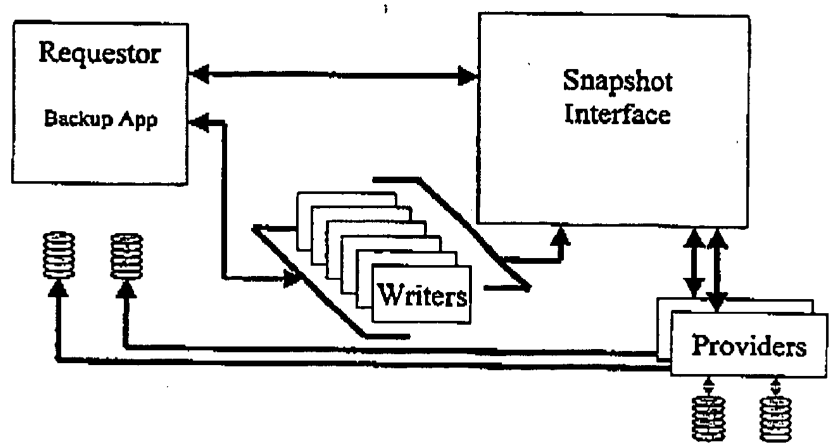
Volume Name	A Win32 definition – the name of a volume in the **\\?\Volume{*GUID*}** syntax that was introduced in Windows 2000.
Volume Snapshot Service	The Windows service that coordinates requestors, providers, and writers. The Volume Snapshot Service is abbreviated as VSS.
Writer	Any application that stores persistent information on one or more volumes, and which participates in snapshot synchronization. Typically this might be a database (e.g. SQL Server) or system service (e.g. Certificate Server).

Overview

An overview of the Volume Snapshot Service (VSS) and its cooperating components is shown on the page following. Summarize the roles:

- Providers own the snapshot data and instantiate the snapshots.
- Writers are applications that are changing data and participate in the snapshot synchronization process.
- Requestors initiate the creation and destruction of snapshots. Our design is focused on the scenario where the Requestor is a backup application.
- VSS provides coordination between these parties.

In addition, there is special support in the Windows file systems flush and hold filesystem data at a common point-in-time across multiple volumes for a snapshot set. This support is known as 'Lovelace', and is collectively implemented by a number of components including VSS, NTFS, and the native volume snapshot driver volsnap.sys.



The Snapshot Provider

Types of Providers

Hardware providers implement the IVssProviderCreateSnapshotSet and IVssHardwareSnapshotProvider interfaces. The first interface is common to all providers and implements the snapshot state sequencing. The second interface is specific to hardware providers and operates on a LUN abstraction.

Providers are implemented as COM components and called out-of-proc by VSS. Providers use provider-specific mechanisms to communicate with the host adapter or external RAID storage subsystem that instantiates the snapshot LUNs. There should be no need for any extra kernel mode component.

Hardware providers operate on LUNs. VSS performs all mappings from original volumes to LUNs, all work to hide or expose snapshots, and orchestrates read-only or read-write access to the snapshot data.

Hardware providers register as VSS_PROV_HARDWARE as defined in vss.idl:

```
typedef enum _VSS_PROVIDER_TYPE
{
        VSS_PROV_UNKNOWN                    = 0,
        VSS_PROV_SYSTEM                     = 1,
        VSS_PROV_SOFTWARE                   = 2,
        VSS_PROV_HARDWARE                   = 3,
} VSS_PROVIDER_TYPE, *PVSS_PROVIDER_TYPE;
```

The system provider is the native volsnap.sys snapshot driver. No other providers should register as VSS_PROV_SYSTEM. No provider should register as

VSS_PROV_UNKNOWN.

Provider Registration and DeRegistration

Only VSS calls providers. The provider registers for calls by invoking IVssAdmin::RegisterProvider(). Once registered, the provider will be involved in snapshot management until deregistration via IVssAdmin::UnRegisterProvider().

Snapshot Provider Load and Unload

Providers can be frequently loaded and unloaded. Providers can provide an optional notification interface IVssProviderNotifications that can be used to detect these situations. Providers only need to implement this interface if it is useful for them to catch these events.

The Snapshot Creation Process

The diagram on the following page shows a non-error representation of the message flow between the various parties in the course of creating a snapshot from the perspective of a requestor.

A requestor is the application that initiates the request to create the snapshot. It is typically a backup application. VSS in turn calls the provider when necessary. From the provider's perspective, the requester makes three important requests:

1. The requestor begins the snapshot creation activity by calling StartSnapshotSet(). VSS implements this function, and the main effect is to generate a new GUID that will be used to identify the snapshot set – the SnapshotSetId. The provider is not involved in this step, but the SnapshotSetId is used extensively in all the subsequent steps.
2. For each volume it wishes to snapshot in this set, the requestor calls AddToSnapshotSet(). VSS determines which provider will be used to snapshot the volume. VSS gives preference to hardware providers, then software providers, and finally the system provider.
    - The same provider need not be used for all volumes in a specific snapshot set.
    - For a hardware provider to be selected, the hardware provider must be able to support all LUNs contributing to the specified volume.
    - The first provider (hardware, software, or system) that accepts the offer to snapshot the volume will be the provider for that volume.
    - There is no guarantee on the order in which hardware providers are called.
3. After one or more calls to AddSnapshotSet(), the requestor can ask for the snapshot to be created using the DoSnapshotSet() function. VSS then communicates with a number of system elements in order to create

the snapshot. The DoSnapshotSet() function performs this work asynchronously, and the requestor can either poll or wait for the snapshot creation process to complete

When the process is complete, the requester can query the SnapshotSet to determine if it was successful, and if not, can determine which elements failed.

It is a key goal to minimize the time interval between freeze and thaw of the writer applications. As such, *every* provider must also have the same goal. Providers must asynchronously start all preparation work related to the snapshot in the BeginPrepareSnapshot() function, and then await its completion in the EndPrepareSnapshot() function. A hardware provider that used plexes could start the sync process no later than BeginPrepareSnapshot() and await it's completion in EndPrepareSnapshot().

Overview of Snapshot Creation Processing

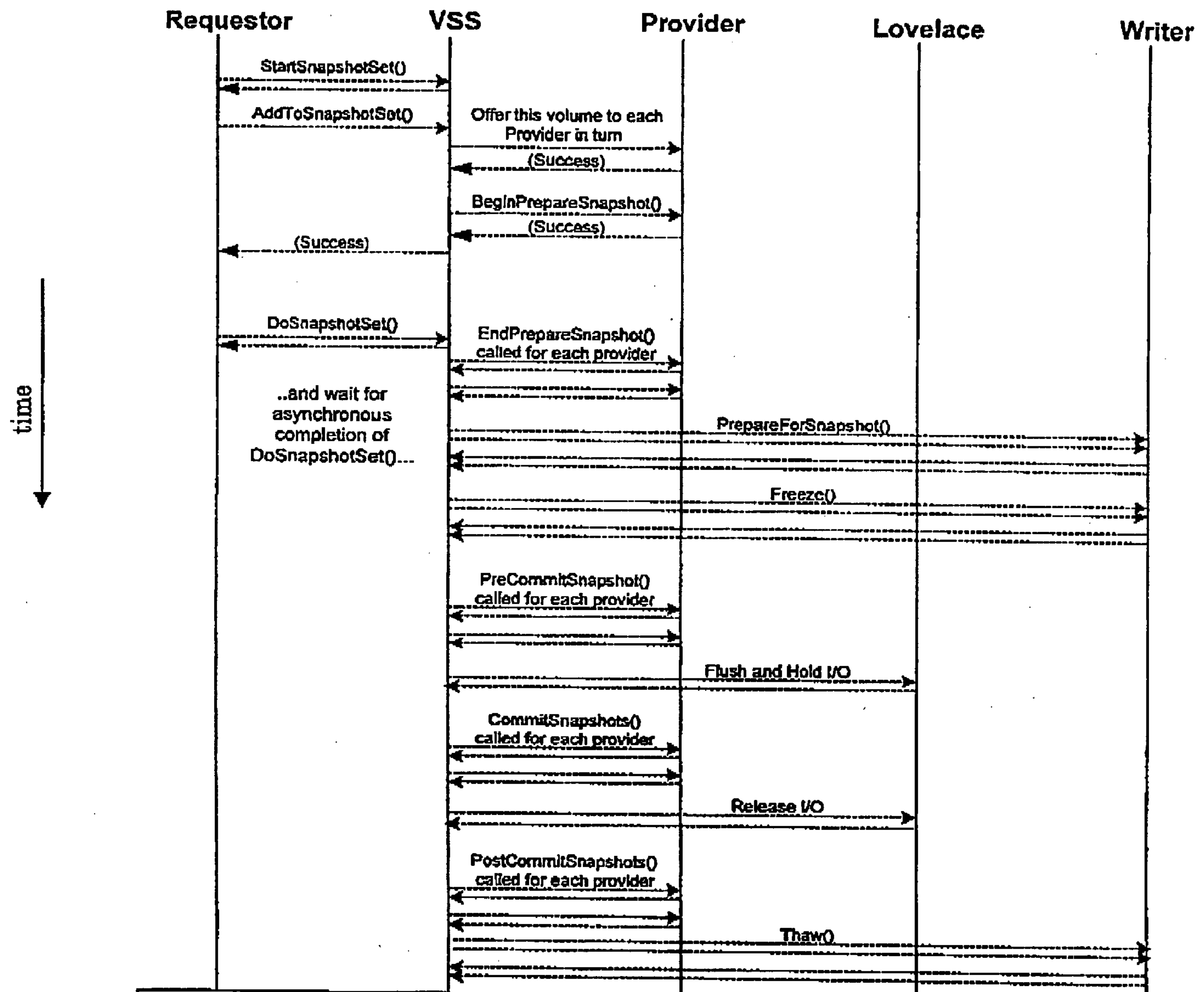


Figure 1. Snapshot Processing overview

Note that the snapshot set is fixed at the invocation of DoSnapshotSet. New volumes cannot be added later; the additional volumes could not have the same point-in-time.

There is a limit of 64 volumes in the snapshot set. A specific volume may map to an entire LUN, a portion of a LUN, entire LUNs, or portions of multiple LUNs. Most configurations will have one volume per LUN, although arbitrary mappings are possible.

There is no limit on the number of snapshot sets or the number of snapshots of an original volume. A provider may define specific limits or dynamically limit based on hardware resources available.

Point-In-time for Legacy Writers

VSS includes special OS kernel support that defines the point-in-time that is common for all volumes in a Snapshot Set. Hardware providers do not need to directly interface with these kernel technologies, since they are invoked as part of the normal snapshot commit processing. However, it is useful to understand the mechanisms involved since it explains the definition of 'point-in-time' for Legacy Applications (applications that have not exposed a 'Writer' interface).

This kernel support for common point-in-time is distributed between the volsnap.sys driver, the filesystems, and VSS. This support is collectively referred to as 'Lovelace' – a codeword that was used during the development of VSS.

1) Before Lovelace is invoked, VSS has already:
   a) Determined which volumes are to be involved in the snapshot
   b) Determined which provider is to be used on each volume.
   c) Frozen the applications that are accepting freeze/thaw messages.
   d) Prepared the providers for the snapshot using PreCommitSnapshot. All providers are just waiting to do the actual split.
2) The point-in-time is then created. VSS concurrently flushes the filesystems on all the volumes that are to be snapshotted.
   a) VSS issues a new IOCTL that flushes the filesystems. That IRP passed down the storage stack to volsnap.sys. Volsnap.sys then holds the non-pagefile write IRPs until step 4) below. Any legacy filesystem (such as RAW) without support for this new IOCTL passes the unknown IOCTL down – where it will is again held by volsnap.sys. Note that on NTFS volumes, the flush also commits the NTFS log.
   b) This suspends all NTFS/FAT metadata activity; the filesystem metadata is cleanly committed.
   c) ***The snapshot instant:*** Volsnap.sys next causes all subsequent non-pagefile write IRPs to be queued on all the volumes that are to be snapshotted.
   d) Volsnap.sys waits for all pending writes on the snapshot volumes to complete[1]. The volumes are now quiesced with respect to writes, and were quiesced at exactly the same moment on each. Note that there are no guarantees about writes to user mapped sections or writes issued between (a) and (b) on filesystems that are do not implement the flush IOCTL (e.g. RAW).

[1] This could take some time if many large I/Os are stacked up deeply in the devices below Lovelace.

3) VSS instructs each provider in take the Snapshot: CommitSnapshots(). The providers should have done all their preparation before so that this is a 'flick the switch' operation.

4) VSS releases all pending write IRPs (including the IRPs that were blocking the filesystems at the conclusion of their commit path) by invoking another new IOCTL passed to volsnap.sys.

5) If the snapshot process was successful, the VSS now:

   a) Issues PostCommitSnapshot to the providers

   b) Issues Thaw to the Writers

   c) Informs the Requestor that the snapshot process has completed

PreCommitSnapshot(), CommitSnapshots() to PostCommitSnapshots() are all time critical. All I/O from applications with writers is frozen from PreCommitSnapshot() to PostCommitSnapshots(); delays affect application availability. All file I/O, including Legacy Application I/O, is suspended during CommitSnapshots().

Providers should do all time critical work should be completed prior to EndPrepareSnapshot().

- CommitSnapshots() should be returned within a few seconds. Lovelace will cancel the IOCTLs that are holding the I/O if the subsequent release is not received within 10 seconds. VSS will fail the snapshot creation process.
- The full sequence of PreCommitSnapshots() to the return of PostCommitSnapshots() must complete within 30 seconds.

During CommitSnapshots() the provider must avoid any non-paging file I/O; such I/O has a very high probability of deadlock. In particular, the provider should not synchronously write any debug or trace logs.

HARDWARE PROVIDER INTERACTIONS AND BEHAVIORS

Hardware providers may support copy-on-write or plex (full mirror copy) snapshots. Resource allocation for snapshots should be "automagic" – reasonable default behavior should occur if the requestor does nothing. The full support for plex resource allocation will be via the post-Whistler Virtual Disk Service. Software copy-on-write providers implement a common management interface for snapshot storage files; a similar interface will be defined when desired by copy-on-write hardware providers.

Hardware providers associated with external SAN RAID subsystems should support transportable snapshots to allow movement between hosts on a SAN. The full support for this will include the post-Whistler Virtual Disk Service and the Fabric Virtualization service. Meanwhile, the provider assumes some of that functionality.

Hardware providers can be stateless across boot epochs. Hardware providers running on multiple machines in a SAN yet managing the same RAID subsystem need not coordinate between providers on a SAN. The coordinator maintains any necessary state. Two kinds of state are recognized:

- State necessary to support data access to the volumes contained on a hardware snapshot. This includes any tagging of a volume as read-only or hidden. This state must be on the hardware LUN and travel with the LUN. This state is preserved across boot epochs and/or device discovery. VSS manages this state.
- State necessary to recognize a specific volume as part of a snapshot set. This state is persisted by VSS in conjunction with the Requestor that originally created the snapshot set.

The Snapshot Creation Process

Creating a snapshot set with a hardware provider proceeds as follows:

1) As the requestor adds each volume to the snapshot set, VSS determines the associated LUNs. Physical device identification information (VDS_LUN_INFORMATION) is retrieved for each.
2) The hardware provider uses the physical device information to determine if the LUNs are supported. This determination must be all or nothing: the same hardware provider must support all LUNs contributing to a specific volume.
3) For each supported LUN, the hardware provider augments VDS_LUN_INFO with any additional information available such as interconnect address.
4) For each volume in the snapshot set, VSS invokes BeginPrepareSnapshot with the same array of VDS_LUN_INFO. Within a single call, all LUNs in the array are unique, however the same LUN may reappear in a subsequent call whenever the same LUN contributes to multiple volumes. The provider must handle that case correctly.
5) Prior to CommitSnapshots, the coordinator tags all affected LUNs. The tag

will cause all volumes on the affected disks to be surfaced on the receiving machine as read-only and hidden.[2] Note that the snapshot LUNs have not yet been separated from the original LUNs; both the original LUNs and the soon-to-be-snapshot LUNs are written at this time.

6) The tag is designed such that in the event of a system crash, the tagged original LUN will not be affected. That is, all volumes on the original (unbroken) LUN will be treated as normal – retaining whatever drive letter/mount points and read/write status.
7) At CommitSnapshots, the snapshot LUNs are broken from the original LUNs. If at all possible, the snapshot LUNs should be masked at the storage subsystem and not simply discoverable by other hosts on the SAN.
8) After CommitSnapshots, VSS removes the tag on all affected LUNs. Since the snapshot LUN has been broken from the original LUN, the snapshot LUN retains the read-only and hidden tags.
9) After PostCommitSnapshots, the Volume Snapshot Set retrieves an array of VDS_LUN_INFO for each newly created snapshot LUN. The provider must provide enough information to allow the newly created snapshot LUN to be moved to another machine. Note that at this time, the snapshot LUN should be inaccessible to this machine; the provider must use mechanism other than simply reading the information from the LUN.
10) VSS appends the following to the XML document describing the snapshot set:
    a) The original LUN VDS_LUN_INFO array
    b) The new snapshot LUN VDS_LUN_INFO array
    c) The disk extents for each volume in the snapshot set
11) The new snapshot LUN VDS_LUN_INFO array is passed to the hardware provider. The provider should unmask the LUN at the storage subsystem and, if possible filter accesses to this machine. The provider should also perform any and all zoning necessary. The provider should not cause a bus rescan (IOCTL_DISK_FIND_NEW_DEVICES). VSS will perform any necessary unmasking at the host and then cause the rescan to allow the LUNs to be discovered by PNP and the volumes to come online.
12) As new volumes are discovered, VSS uses the original LUN VDS_LUN_INFO ARRAY and the disk extents for each volume in the snapshot set to determine which new volume corresponds to which original volume.
13) At this point, all volumes contained on the new snapshot LUNs are mounted hidden and read-only and VSS has a mapping from original volume to snapshot volume.

Post Whistler, VSS will invoke the Virtual Disk Service (and implicitly the Fabric Virtualization Service) to locate, unmask, zone, and online each LUN in the snapshot set. This will be transparent to the provider.

[2] The tag is implemented using hidden sectors on MBR disks or operating system software specific bits in the partition header entry on GPT disks.

The volumes are created as read-only and hidden to minimize confusion. Because there is not a one-to-one mapping of volumes to LUNs, the set of snapshot LUNs may inadvertently include "extra" volumes. When the snapshot is surfaced, those volumes remain hidden from applications.

After creation, a snapshot set may be converted to read-write and/or exposed via a drive letter, mount point, or share.

Transporting Snapshots on a SAN

Snapshot sets may be transported, moved between hosts, on a SAN. The set may be transported one or many times after initial creation.

From the point of view of a provider capable of transporting LUNs between hosts, there is little or no difference between a non-transportable and a transportable snapshot set.

Transportable snapshot sets must be created with the TRANSPORTABLE context attribute. All volume in the set must be transportable. In step 2) above, the provider must not only support the LUN, but also be able to transport it to accept the LUN. Creation of a transportable snapshot set concludes at step 10) above when VSS records the LUN information in the XML document.

On the receiving machine, the requestor imports the snapshot set. This method uses the XML document describing the snapshot set as input. Importing proceeds by:

1) VSS constructs an array of snapshot LUN VDS_LUN_INFO from the XML document.
2) That array is passed to the hardware provider. The provider should unmask and/or zone all LUNs, but need not cause a bus rescan (IOCTL_DISK_FIND_NEW_DEVICES). VSS will perform any unmasking at the host and then cause the rescan to allow the LUNs to be discovered by PNP and the volumes to come online.
3) As new volumes are discovered, VSS uses the original LUN VDS_LUN_INFO ARRAY and the disk extents for each volume in the snapshot set to determine which new volume corresponds to which original volume.
4) At this point, all volumes contained on the transported LUNs are mounted hidden and read-only and VSS has a mapping from original volume to snapshot volume.

Note the similarities between steps 11 through 13 in the single machine case and steps 2 through 4 when transporting.

Required Behaviors

A snapshot provider must have the following behavior to ensure requestor compatibility. At run time, a backup application or other requestor that uses snapshots can function correctly without any knowledge of specific provider implementation details.

Automagic Resource Management

It must be possible for the requestor simply to add a volume to a snapshot set. The provider must include an automatic rule for allocating drive space for the implied plex or copy-on-write snapshot LUN. In the absence of such a rule, the provider must never agree to support the volume.

Plex providers may include a provider-specific interface for resource management. That API is only temporary; post-Whistler binding a plex to a LUN will be done via the Virtual Disk Service.

The hardware provider must not include a provider-specific API for managing this. Software providers implement a common management API; a common hardware provider API will be added when necessary.

Create Snapshot Volumes as Read-only and Hidden

VSS tags each volume on each affected LUN such that the resulting snapshot plex will be hidden and read-only when detected by a Whistler machine. Drive letters and/or mount points are not automatically assigned.

Hardware providers need do nothing to comply with this requirement.

Hardware Snapshots not Supported on Wolfpack

Wolfpack cannot accommodate LUNs with duplicate signatures and partition layout. The snapshot LUNs must be transported to a host outside the cluster.

Snapshots Containing Dynamic Disks must be Transported

The native support for dynamic disks cannot accommodate LUNs with duplicate signatures and configuration database contents. The snapshot LUNs must be transported to a different host. VSS enforces this.

When transporting dynamic disk LUNs to a new host, it is best that at least one dynamic disk already exist on the receiving host. This ensures that the disk group identifiers will be unique to both machines.

Garbage Collection of Non-Persistent Snapshots

VSS is designed to automatically delete non-persistent snapshots when the requestor releases its interface; this includes the case where that release is the result of a crash of the requestor. The same garbage collection should occur in

the event of VSS crash or restart.

This affects hardware providers in one and only one way: if a snapshot has been committed but the VDS_LUN_INFO associated with the new snapshot LUN has not yet been passed to VSS, any data on that LUN must remain inaccessible. A plex LUN may be resynchronized with the original LUN or simply treated as unallocated. Copy-on-write space must be reclaimed.

Providers are Out-of-Proc

All providers must be implemented as out-of-proc COM components. This is because there is no performance reason to have them as in-proc components, and because this helps to isolate the coordinator from implementation issues in third-party providers.

Time Critical Operations

As noted previously, long operations such as syncing mirrors, should be initiated in BeginPrepareSnapshot(). This function should return immediately, but perform the required preparation work asynchronously.

EndPrepareSnapshot() serves as a 'rendezvous' function; the provider can wait synchronously in this method for completion of the preparation work that was started by BeginPrepareSnapshot().

Providers must not add delays in PreCommitSnapshot(), CommitSnapshot(), or PostCommitSnapshot() since applications are frozen at this time.

Careful I/O during CommitSnapshots

Hardware providers should not need to do any I/O to the affected volumes during CommitSnapshots. If any I/O is required, providers must take great care with any I/O issued to an original volume during the CommitSnapshots() function.

During CommitSnapshots(), VSS is using the 'Lovelace' drivers to block I/O to the original volumes being snapshotted. If the provider uses synchronous file I/O to a volume which is in the snapshot, then that I/O will be blocked and the snapshot commit process will be deadlocked. The Lovelace timeout will break this deadlock after 10 seconds, but this will cause the snapshot to fail. Providers must not provoke such a deadlock and failure. If they do deadlock in this way they will be considered faulty and unsupportable.

If I/O must be attempted, it must be performed asynchronously. The I/O will not complete until after all providers have returned from their CommitSnapshots() method. In general, it is better to perform such I/O in the later PostCommitSnapshots() call.

State Transitions

The state model in a snapshot provider is greatly simplified by the fact that

snapshot creation for a snapshot set is serialized all the way from StartSnapshotSet() through to the completion of DoSnapshotSet().

The following table lists the valid state transitions for a snapshot.

Table. Snapshot state transitions

Initial State	Operation	Resultant state
VSS_SS_UNKNOWN	Enter BeginPrepareSnapshots()	VSS_SS_PROCESSING_PREPARE
VSS_SS_PROCESSING_PREPARE	Leave BeginPrepareSnapshots()	VSS_SS_PREPARING
VSS_SS_PREPARING	Leave EndPrepareSnapshots()	VSS_SS_PREPARED
VSS_SS_PREPARED	Enter PreCommitSnapshots()	VSS_SS_PROCESSING_PRECOMMIT
VSS_SS_PROCESSING_PRECOMMIT	Leave PreCommitSnapshots()	VSS_SS_PRECOMMITED
VSS_SS_PRECOMMITED	Enter CommitSnapshots()	VSS_SS_PROCESSING_COMMIT
VSS_SS_PROCESSING_COMMIT	Leave CommitSnapshots()	VSS_SS_COMMITED
VSS_SS_COMMITED	Enter PostCommitSnapshots()	VSS_SS_PROCESSING_POSTCOMMIT
VSS_SS_PROCESSING_POSTCOMMIT	Leave PostCommitSnapshots()	VSS_SS_CREATED
Any	AbortSnapshots()	VSS_SS_ABORTED

Error Handling

VSS allows many snapshots to exist at once, but it only allows one snapshot set to be proceeding between StartSnapshotSet through DoCommitSnapshot().

No Partial Commit

If any provider fails a snapshot on any volume or LUN in the snapshot set, then the creation of the entire snapshot set is aborted. There is no way to avoid this behavior. It is defined this way in order to simplify support and behavioral issues associated with partial failure semantics.

Reporting Fault Conditions

In the event of any hardware provider error or fault condition, the provider must log an error to the system event log. This includes, but is not limited to, any

provider-specific error when creating or importing a Snapshot Set or resource allocation failure for copy-on-write snapshot after creation.

Unsupported Scenarios

Snapshot providers may only snapshot a volume if they are able to snapshot all LUNs contributing to the volume. Anything else would require a level of coordination between providers. Note that this specifically excludes the case where a volume has been grown by concatenation and exists partly on a LUN managed by hardware-based provider A, and partly on a LUN managed by hardware-based Provider B or when the volume is striped between such LUNs.

HARDWARE PROVIDER INTERFACES AND METHODS

Introduction

The Hardware Snapshot Provider interfaces are IVssAdmin, IVssProviderNotifications, IVssProviderCreateSnapshotSet, and IVssHardwareSnapshotProvider.

IVssAdmin

The **IVssAdmin** interface is implemented by VSS, and manages the list of registered providers.

Methods

Name	Description
QueryProviders	Queries all registered providers.
RegisterProvider	Registers a new snapshot provider.
UnregisterProvider	Unregisters an existing provider.

IVssProviderNotification

Snapshot providers can be frequently loaded and unloaded. To detect this, providers can provide an optional Notification Interface IVssProviderNotifications. Implementation is optional; providers only need to implement this interface if it is useful for them to catch these events.

The **IVssProviderNotifications** interface is implemented by the provider and is used by VSS to notify the provider about specific events. Every provider must support this interface. This interface must be accessible using **IVssHardwareSnapshotProvider::QueryInterface.**

Methods

Name	Description
OnLoad	Called by VSS to notify the provider that it was just loaded.
OnUnload	Called by VSS to notify the provider that it will be unloaded.

IVssProviderCreateSnapshotSet

The **IVssProviderCreateSnapshotSet** interface contains the methods during snapshot creation. All providers must support this interface; the interface is common to software and hardware providers.

Methods

Name	Description
EndPrepareSnapshots	Ensure all LUNs in the snapshot set are prepared
PreCommitSnapshots	Ensure that the provider is ready to quickly commit the prepared LUNs. This

	happens immediately before the Flush-and-hold writes, but while applications are in their frozen states.
CommitSnapshots	Quickly commit all LUNs in this provider. Special restrictions exist on what operations the Provider may perform during this call.
PostCommitSnapshots	Called after all the snapshots have been committed. This happens immediately after the release-writes to the I/O subsystem, but while applications are in still in their frozen states.
AbortSnapshots	Ends the prepared snapshots in this provider. This includes all non-committed snapshots and any pre-committed ones.

IVssHardwareSnapshotProvider

Each hardware provider must implement the **IVssHardwareSnapshotProvider** interface. The COM class that implements this interface is specified by the administrator in **IVssAdmin::RegisterProvider** at registration time.

Methods

Name	**Description**
AreLunsSupported	Allows VSS to determine if this hardware provider can snapshot the LUNs that contribute to a specific original volume. The provider also updates the VDS_LUN_INFO structure.
BeginPrepareSnapshot	Adds LUNs to the snapshot set.
GetTargetLuns	Retrieves the hardware identification information for each new created LUN
LocateLuns	Performs any necessary RAID subsystem unmasking and/or zoning to allow a snapshot LUN to be discovered by this machine.
OnLunEmpty	Notifies the provider that a LUN that previously contained snapshot no longer contains data of interest.

IVssAdmin: Registration of Snapshot Providers

A provider registers with VSS via **IVssAdmin::RegisterProvider():**

```
STDMETHODIMP IVssAdmin::RegisterProvider(
    IN      VSS_ID      ProviderId,
    IN      CLSID       ClassId,
    IN      VSS_PWSZ    pwszProviderName,
    IN      VSS_PROVIDER_TYPE eProviderType,
    IN      VSS_PWSZ    pwszProviderVersion,
    IN      VSS_ID      ProviderVersionId
    )
```

IVssAdmin::UnRegisterProvider() deregisters the provider and removes it and any snapshots instantiated by the provider from snapshot management.

The ProviderId is a GUID that uniquely and persistently identifies the Provider. For example the volsnap.sys provider is defined as:

```
const GUID VSS_SWPRV_ProviderId = { 0xb5946137, 0x7b9f, 0x4925, { 0xaf,
0x80, 0x51, 0xab, 0xd6, 0xb, 0x20, 0xd5 } };
```

Once defined, the ProviderId should remain the same; this is true even when the software revision is updated. The only reason for changing a provider GUID is when the provider functionality changes and both providers might reasonably be active on the same system.

IVssProviderCreateSnapshot: Creating Snapshots

The **IVssProviderCreateSnapshotSet** interface contains the methods used during snapshot creation. All providers must support this interface; the interface is common to software and hardware providers.

For all methods, a successful return indicates that processing for any and all LUNs in the snapshot set was successful.

IVssProviderCreateSnapshot::EndPrepareSnapshots

This method will be called once for the complete snapshot set. After this is called, there will be no more BeginPrepareSnapshot calls. This method is intended as a rendezvous where the provider can wait for any snapshot preparation work to complete.

```
HRESULT EndPrepareSnapshots(
    [in]    VSS_ID          SnapshotSetId,
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier

IVssProviderCreateSnapshot::PreCommitSnapshots

The **PreCommitSnapshots** method is called prior to snapshot commit. It should

be used to prepare all snapshots in this SnapshotSet for committing by the subsequent CommitSnapshots() method. While this is called, applications have been frozen (but the I/O subsystem is not yet blocking filesystem I/O) so the provider should attempt to minimize the amount of time spent in this method.

```
HRESULT PreCommitSnapshots(
      [in]    VSS_ID           SnapshotSetId,
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier.

IVssProviderCreateSnapshot::CommitSnapshots

The **CommitSnapshots** method is called at the defined instant at which the snapshots should be taken. For each prepared LUN in this snapshot set, the provider shall perform whatever work is appropriate in order to persist the point-in-time LUN contents. While this method is called, both applications and the I/O subsystem are quiesced so the provider must attempt to minimize the amount of time spent in this method. As a general rule, a provider should spend less than 1 second in this method.

In addition, since the I/O system is quiesced at this time, the provider shall take great care not to initiate any I/O that may deadlock the system - for example debug/tracing I/O by the method or by any methods it has invoked (Note that VM-oriented file or paging I/O will not be frozen at this time).

```
HRESULT CommitSnapshots(
      [in]    VSS_ID           SnapshotSetId,
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier.

IVssProviderCreateSnapshot::PostCommitSnapshots

The **PostCommitSnapshots** method is called after all providers involved in the snapshot set have succeeded with CommitSnapshots, and VSS has released the 'Lovelace' lock on the system I/O. Note that applications are still frozen at this time.

This method is an opportunity for the provider to provide additional cleanup work after the snapshot commit. Note that `lSnapshotCount` should not be needed by hardware providers but is necessary for software providers.

```
HRESULT PostCommitSnapshots(
      [in]    VSS_ID           SnapshotSetId,
      [in]    LONG             lSnapshotCount
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier.

lSnapshotCount
[in] Count of snapshots in the snapshot set.

IVssProviderCreateSnapshot::AbortSnapshots

The **AbortSnapshots** method aborts prepared snapshots in this provide. This includes all non-committed snapshots and pre-committed ones.

```
HRESULT AbortSnapshots(
    [in]    VSS_ID          SnapshotSetId
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier.

IVssHardwareSnapshotProvider: Managing LUNs and Volumes

The **IVssHardwareSnapshotProvider** interface contains the methods used by VSS to map volumes to LUNs, discover LUNs created during the snapshot process, and transport LUNs on a SAN. All hardware providers must support this interface.

VDS_LUN_INFO

VDS_LUN_INFO structure contains all hardware properties that can be used to locate a LUN.

VSS initializes the fields from the SCSI 0x00, 0x80, and 0x83 commands for all LUNs that contribute to a snapshots set. The provider initializes any interconnect specific addresses for any such LUNs and/or corrects any omissions.

For all LUNs created by committing a snapshot, the provider initializes all fields. This allows the newly created LUNs to be located by Windows software both on the original machine and/or any other machine in a SAN.

```
typedef struct _VDS_LUN_INFO
    {
    ULONG m_version;

    // The SCSI-2 device type
    BYTE m_DeviceType;

    // The SCSI-2 device type modifier (if any) - this may be zero
    BYTE m_DeviceTypeModifier;
```

```
// Flag indicating whether the device can support multiple
// outstanding commands.  The actual synchronization in this
// case is the responsibility of the port driver.
BOOL m_bCommandQueueing;

// Contains the bus type (as defined above) of the device.  It
//  should be used to interpret the raw device properties at
//  the end of this structure (if any)
VDS_STORAGE_BUS_TYPE BusType;


// vendor id string.  For devices with no such ID
// this will be zero
[string] char *m_szVendorId;

// device's product id string.  For devices with no such ID
// this will be zero
[string] char *m_szProductId;

// zero-terminated ascii string containing the device's
// product revision string.  For devices with no such string
// this will be zero
[string] char *m_szProductRevision;

// zero-terminated ascii string containing the device's
// serial number.  For devices with no serial number
// this will be zero
[string] char *m_szSerialNumber;

// device id descriptor
VDS_STORAGE_DEVICE_ID_DESCRIPTOR m_deviceIdDescriptor;

 // number of interconnects
 ULONG cInterconnects;

 // array of interconnects
 [size_is(cInterconnects)] VDS_INTERCONNECT *rgInterconnects;

 } VDS_LUN_INFO;

typedef struct _VDS_INTERCONNECT
    {
    // address type
    VDS_INTERCONNECT_ADDRESS_TYPE m_addressType;
```

```
    // size of address
    ULONG m_cbAddress;

    // address
    [size_is(m_cbAddress)] BYTE *m_pbAddress;
    } VDS_INTERCONNECT;
```

Notes:

- All disk or LUN identification structures are defined by the Virtual Disk Service (VDS). The Volume Snapshot Service and Fabric Virtualization Service use these same definitions.
- The VDS_STORAGE_DEVICE_ID_DESCRIPTORS directly corresponds to the return from context page 0x83.
- VDS_INTERCONNECT_ADDRESS_TYPE is an enumeration of recognized interconnect addressing schemes and includes, but is not limited to, FCFS, FCPH, FCP3, MAC (iSCSI), and SCSI.

IVssHardwareSnapshotProvider::AreLunsSupported

This method will be called for each snapshot that is added to the snapshot set. Prior to invoking this method, VSS determines the LUNs that contribute to the LUN.

For a specific volume, each LUN can contribute only once; a specific LUN may contribute to multiple volumes. VSS does no tracking of LUNs. The same LUN will never appear more than once in a single call, but may reappear on subsequent calls. Consider the case of two snapshot volumes: D: and E:. D: is an extended volume contained on LUNS 1 and 2. E: is a simple volume contained on LUN 2. If both volumes are added to the same snapshot set, LUN 2 will appear on subsequent calls.

Prior to returning success, the provider updates the VDS_LUN_INFO structure with the LUN interconnect address(es) and any additional information to ensure later recognition of the LUN.

```
HRESULT AreLunsSupported (
    [in]    LONG    lLunCount,
    [in, out, size_is(lLunCount)]
            VDS_LUN_INFO *pLunInformation
);
```

Parameters

lLunCount

[in] Number of LUNs contributing to this snapshot volume

pLunInformation
[in, out] Array of VDS_LUN_INFO for each LUN contributing to the snapshot volume.

IVssHardwareSnapshotProvider::BeginPrepareSnapshot

This method will be called for each snapshot that is added to the Snapshot set.

```
HRESULT BeginPrepareSnapshot(
      [in]    VSS_ID          SnapshotSetId,
      [in]    VSS_ID          SnapshotId,
      [in]    LONG            lLunCount,
      [in, unique, size_is(lLunCount)]
              VDS_LUN_INFO *rgLunInformation
);
```

Parameters

SnapshotSetId
[in] Snapshot set identifier.

SnapshotId
[in] Name of the volume the snapshot is to be created on.

lLunCount
[in] Number of LUNs contributing to this snapshot volume

rgLunInformation
[in] Array of VDS_LUN_INFO for each LUN contributing to the snapshot volume

IVssHardwareSnapshotProvider::GetTargetLuns

This method will be called once after PostCommitSnapshots for the complete snapshot set. Identifying information for each newly created LUN is returned to VSS. That information must include not only the device attributes (eg serial number), but also any and all network addresses.

```
HRESULT GetTargetLuns(
      [in] LONG        lLunCount,
                       VDS_LUN_INFO *rgSourceLuns,
      [out, size_is(lLunCount)]
                       VDS_LUN_INFO *rgDestinationLuns
);
```

Parameters

lLunCount
[in] Number of LUNs contributing to this snapshot volume

rgSourceLuns
[in] Array of VDS_LUN_INFO for each LUN contributing to the snapshot volume snapshot set identifier.

rgDestinationLuns
[out] Array of VDS_LUN_INFO for each new LUN created during snapshot processing. There should be a one-to-one correspondence between each element of rgSourceLuns and rgDestinationLuns.

IVssHardwareSnapshotProvider::OnLunEmpty

This method is called whenever VSS determines that snapshot LUN contains no interesting data. All snapshots have been deleted (which also causes deletion of the volume). The LUN resources may be reclaimed by the provider and reused for another purpose.

Note that OnLunEmpty is called on a best effort basis. VSS invokes the method ONLY when the LUN is guaranteed to be empty. There may be many cases where the LUN is empty but VSS is unable to detect this. An example of this case is when a snapshot LUN is moved to a different host but not actually transported or imported by VSS. That LUN appears as any other LUN and volumes can be simply deleted via Disk Management without any notification of VSS.

```
HRESULT OnLunEmpty (
    [in, unique]  VDS_LUN_INFO *pInformation
);
```

Parameters

pInformation
[in] The VDS_LUN_INFO for an empty LUN

IVssHardwareSnapshotProvider::LocateLuns

This method will be called once when a snapshot set is transported between machines. The provider is responsible for any unmasking at the hardware and any necessary switch zoning. VDS_LUN_INFO passed to the provider is exactly that received by VSS at GetTargetLuns.

Immediately after this method completes, VSS will perform any host-based unmasking and invoke IOCTL_DISK_FIND_NEW_DEVICES. This causes any exposed LUNs to be discovered by PNP. Note that host-based masking is a post-Whistler feature.

```
HRESULT LocateLuns(
    [in]        LONG    lLunCount,
    [in, unique, size_is(lLunCount)]
                VDS_LUN_INFO *rgSourceLuns
```